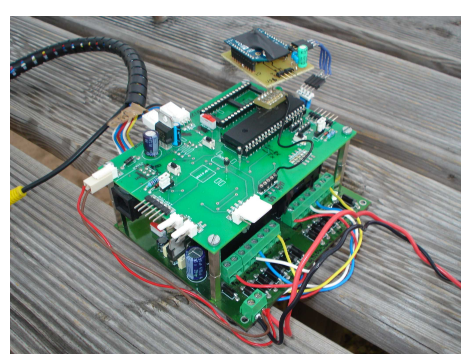
Note d'application

Station sol
Version 2

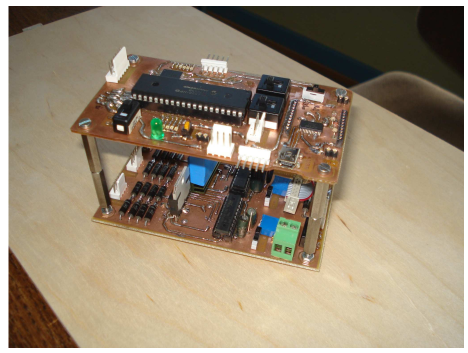
Stage Technicien Juin-Juillet 2009



Antenne Vidéo motorisée de la station sol



Station sol - Version 1



Station sol - Version 2

Pierre-Alain Vercruysse, GE4

Sommaire

I - Descriptif de la station sol :

1) Rôle de la station sol :

Figure 1 : Antenne motorisée

L'objectif principal de la station sol est de faire un suivi du drone par une antenne (*cf. Figure 1*). Le drone étant équipé d'un émetteur vidéo, la station sol récupère le flux vidéo par un module de réception posé sur le mat, et retransmet ce flux directement par USB grâce au module "Grabshow". Par la suite, il serait intéressant d'intégrer une seconde antenne qui ferait la réception et l'émission des trames Xbee Sol-Air d'abord puis par exemple Sol-Sol (pour une télécommande ou le Joystick). La station sol a par ailleurs d'autres rôles. Voici un listing de ses taches :

- Acquisition de la trame GPS (@#) du drone pour effectuer le suivi d'antenne.
- Acquisition de la trame GPS (@#) du drone pour un ordinateur branché sur la station sol (USB).
- Acquisition de la trame PID (@P) de l'ordinateur (USB) qui permet de modifier les PID en vol.
- Acquisition des consignes Joystick (en mode filaire actuellement).
- Envoi des consignes Joystick et de la trame PID.
- Envoi de certaines données à afficher sur les lunette 3D.

Une première station avait été développée lors d'un projet MIQ4 en 2008/2009 (*cf. Figure 2*). Ce travail avait débouché sur la réalisation d'une double carte commande et puissance. Cette carte qui est tout à fait fonctionnelle présente cependant quelques défauts. Une nouvelle version de ces cartes a donc été réalisée.

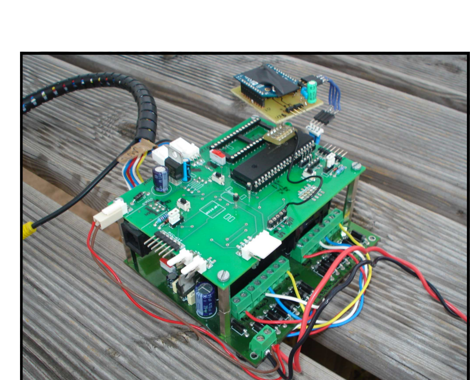


Figure 2 : Station sol - Version 1

2) Schéma du drone en vol :

Figure 3 : Schéma du drone en vol

3) Diagramme de fonctionnement :

La carte s'appuie sur 2 microcontrôleurs PICs 18F. L'un va se charger de motoriser l'antenne afin d'orienter celle-ci toujours dans la direction du drone. Le second PIC va être un aiguilleur d'information pour l'incrustation vidéo et l'envoi de la trame pour la correction des PIDs.

Le Xbee transmet la trame GPS aux deux PICs ainsi qu'au PC (interfacé en USB) contenant l'interface graphique. Le PC va ensuite renvoyer une trame PID contenant les corrections à apporter en vol sur les PID, ainsi que les données qu'il faudra afficher lors de l'incrustation vidéo. Le PIC aiguilleur va donc récupérer la trame GPS ainsi que la trame PIDs. Il va par ailleurs recevoir des consignes Joystick. À partir de ce flux d'informations, il va construire la trame PID (@P) envoyée par Xbee au drone. Il va également afficher sur les lunettes 3D les informations souhaitées (Altitude, Vitesse …) par le biais du module de surimpression vidéo (incrustation vidéo). Le switch SW permet de reconfigurer le Xbee par USB lors d'un déréglage de celui-ci.

Voici un diagramme de fonctionnement de la station :

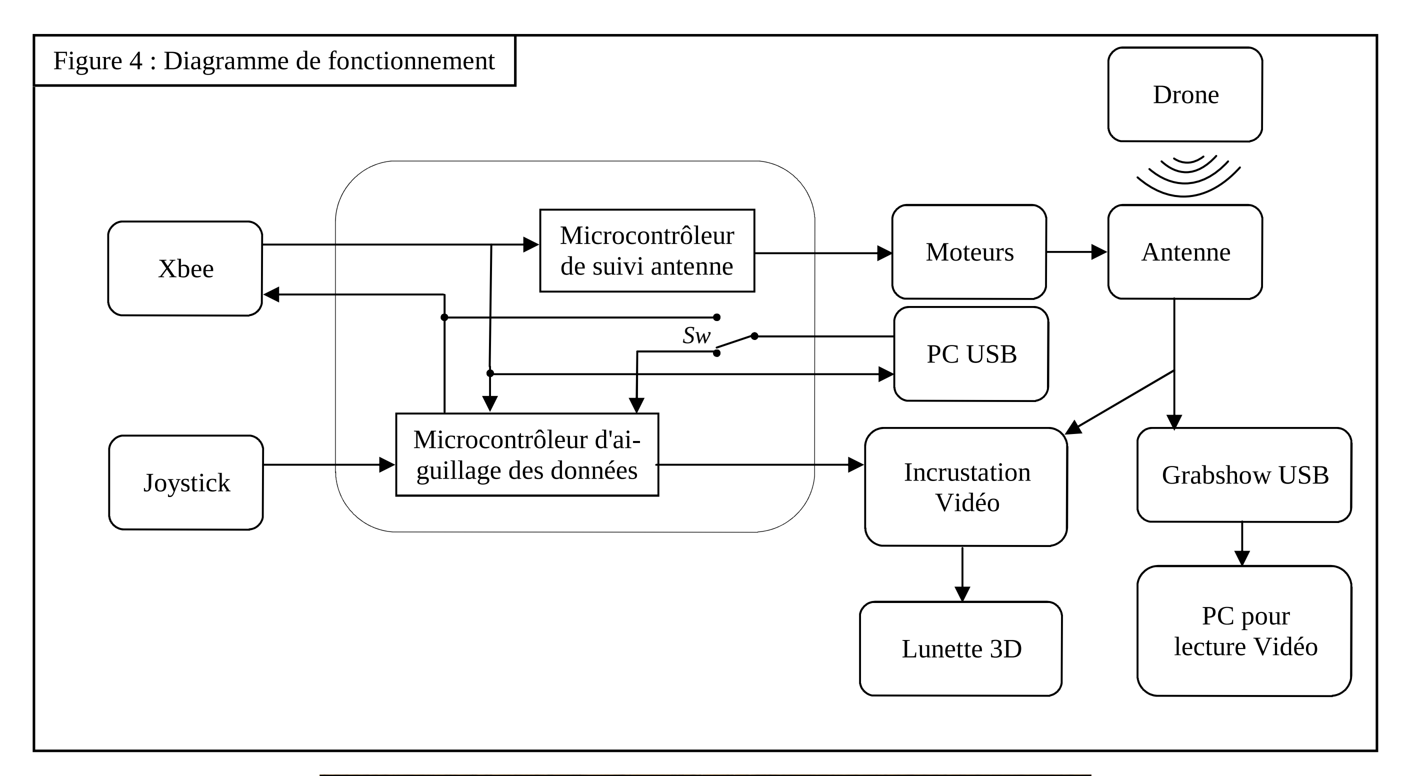




Figure 5 : Récepteur Vidéo et Grabshow USB

4) Coordonnées GPS :

Les coordonnées GPS sont envoyées par la trame GPS (@#). Elles ont pour unité les degrés et se composent de la longitude (axe "X") et de la latitude (axe "Y") ayant pour origine respectivement le méridien de Greenwich et l'équateur. La Terre étant approximée à une sphère de rayon R = 6374 km, par un simple produit en croix, on peut trouver une équivalence entre les degrés et les mètres : $360° = 2\pi.R$

Figure 6 : Coordonnées GPS

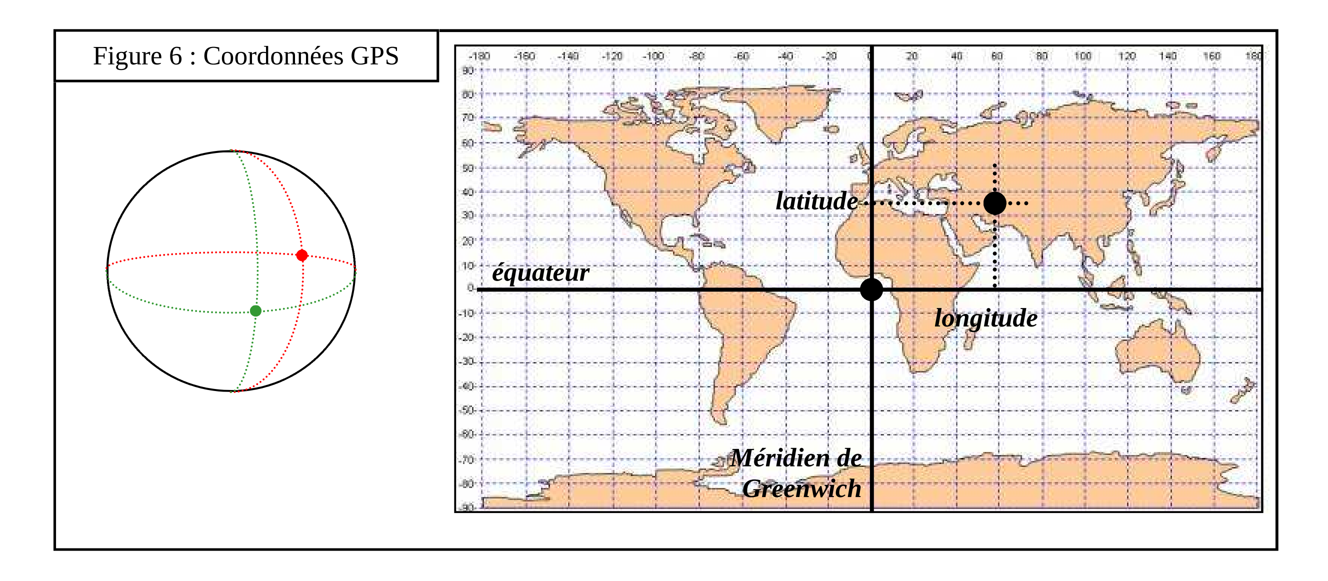




Figure 7 : Impression écran issu du site "gpsfrance.net"

Quelle est la précision du GPS ? Pour une même localisation, comment fluctuent les coordonnées ?

II - Descriptif de la carte électronique :

1) Contexte de la station sol :

La carte électronique s'inspire logiquement de l'architecture souhaitée (détaillée en Figure 3 sur la page 3). La structure de la carte suit celle de la précédente version. La station sol s'articule autour de deux microcontrôleurs PIC : un 18F4520 pour la gestion de la commande des moteurs, et un 18F6520 qui s'occupera de l'aiguillage des données entre le Xbee, le PC, les lunettes 3D ainsi que le Joystick.

1) a - Choix des microcontrôleurs :

Les PICs retenus pour cette deuxième version de la carte restent les mêmes que pour la première version. Il faut cependant noter que le PIC 18F4520 est sans doute surdimensionné. En effet, un grand nombre d'entrées/sorties restent inutilisées. Un PIC comme le 18F2221 (28 pins au lieu de 40) pourrait remplir ce rôle. Cependant, pour éviter de s'éloigner de la carte V1, le même PIC a été gardé. En ce qui concerne le PIC échangeur (18F6520), il a été choisi pour ses deux ports UART. Ce PIC a été parfaitement choisi puisque seuls deux PICs possèdent 2 UART (communication série) et 4 CCP (Capture Compare). Cependant, ce PIC n'existe que au format CMS TQFP(64pins), ce qui est un peu embêtant pour les soudures non professionnelles.

1) b - Organisation de la carte :

Voici un schéma simplifié de l'organisation de la station sol :

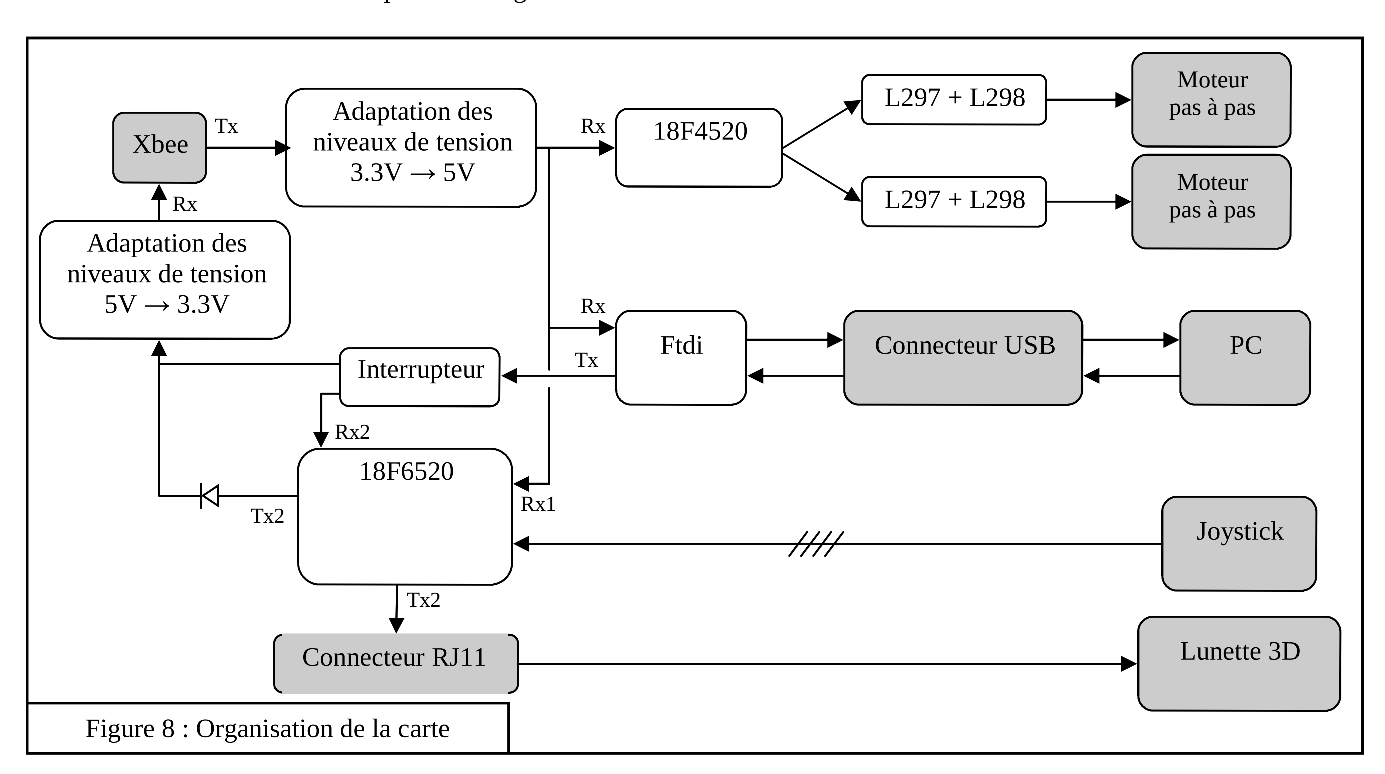


Figure 8 : Organisation de la carte

Ce schéma ne fait intervenir que les composants les plus importants. À ces composants s'ajoutent des réseaux de 6 LEDs (4 vertes, 2 rouges) pour chaque PIC ainsi que des Interrupteurs, un Bouton Poussoir, des connecteurs Pic-Kit, un connecteur pour le Compas (Projet annexe), etc…

2) Détails des différents composants de la station sol :

2) a - Cablage du FTDI :

Concernant le FTDI : il s'agit d'un composant permettant la conversion de signaux UART en signaux USB directement lisibles par un ordinateur grâce à un logiciel comme X-CTU. L'ordinateur (qui est en fait l'interface graphique) récupère donc les informations de la trame GPS (coordonnées, altitude, vitesse, etc…), et renvoie une trame PID (corrections des PID, sélection des waypoints, affichage d'informations sur l'incrustation vidéo …) par le biais du 18F6520. Ce dernier, le PIC d'aiguillage va récupérer les consignes Joystick, les PID et va les retransmettre par Xbee de façon structurée. Il a également été convenu que le PC enverrai une première trame indiquant quelles indications devaient être imprimées sur les lunettes (cf. explication sur le code de ce PIC). Le pic d'aiguillage se chargera donc d'envoyer régulièrement les informations à afficher sur la lunette.

Le câblage de ce composant est indiqué sur le schéma ci-dessous. Les diodes D17 et D18 permettent d'éviter un conflit entre les alimentations de l'USB et celle de la carte. En effet, si ces diodes n'étaient pas là, il y aurait le 5V USB qui irait directement sur le 5V régulé de la batterie. Dans tous les cas, le FTDI sera alimenté par une seule des deux tensions (la plus grande), et les deux alimentations resteront séparées l'une de l'autre.

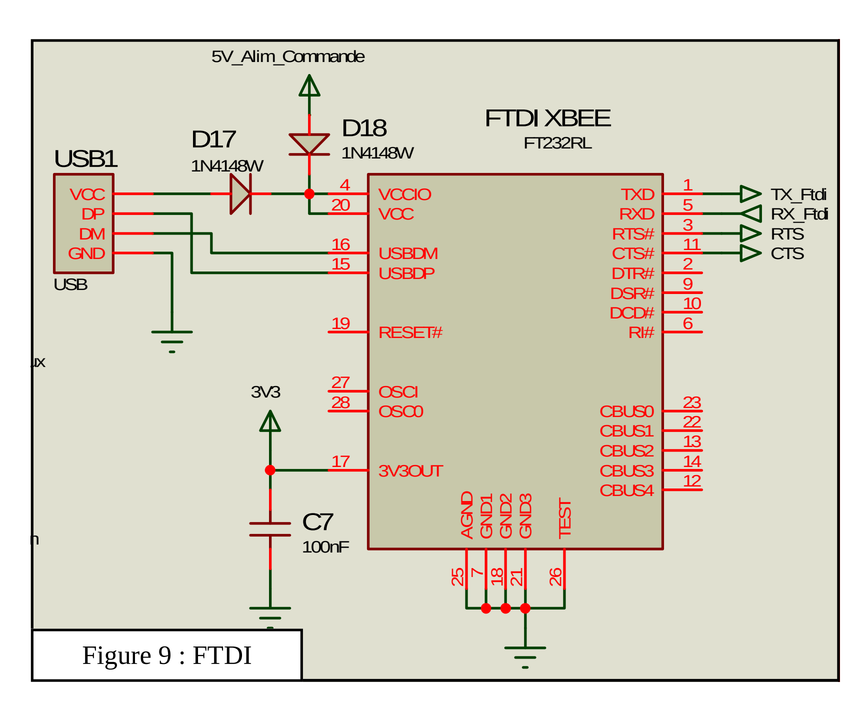
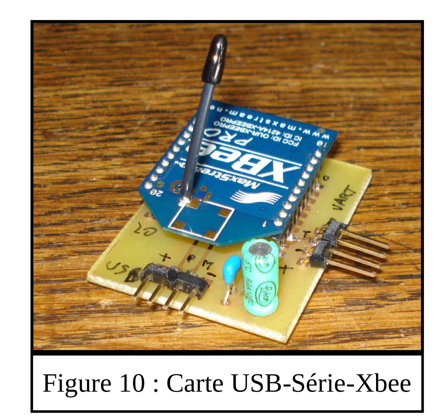


Figure 9 : FTDI

Le FTDI est un composant très pratique. Grâce à ce composant, il a été possible de remplacer la grosse puce « USBMOD3 » précédemment installée sur la station sol V1. En regardant bien, l'USBMOD3 intègre lui-même un FTDI. Par ailleurs, le FTDI créé une alimentation de 3,3V que l'on peut réutiliser sur la carte pour alimenter le Xbee. On évite ainsi l'ajout d'un composant qui remplirait cette fonction. Cependant, cette alimentation 3,3V fluctue entre 3V et 3,6V avec un courant admissible de 50mA. Le Xbee PRO pouvant consommer un peu plus, il faudra donc faire attention à ce paramètre là. En effet, sur la datasheet, il n'est pas indiqué clairement les courants consommés. On peut cependant se référer aux valeurs des courants RX et TX à 3,3V (respectivement 55mA et 215mA). Lors de plusieurs essais, la carte de test "USB-Série-Xbee" (cf. Figure 10) a parfaitement bien fonctionné.


Figure 10 : Carte USB-Série-Xbee

2) b - Câblage du Xbee :

Les niveaux fournis par le Xbee ne sont pas directement des signaux compatibles à ceux des PIC. Il fallait donc adapter la l'entrée IN du Xbee (5V→3.3V) ainsi que la sortie OUT du Xbee (3.3V→5V) :

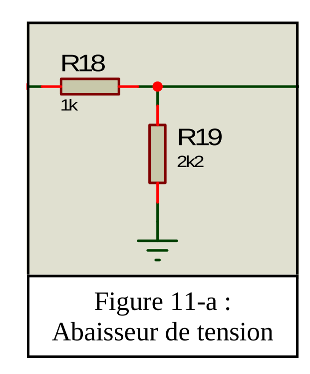


Figure 11-a : Abaisseur de tension

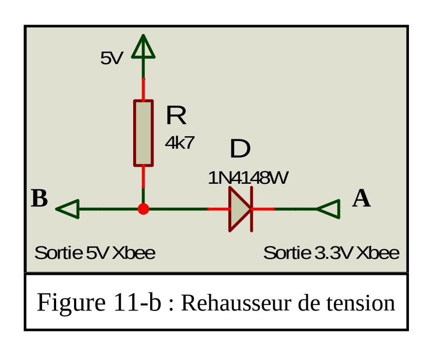


Figure 11-b : Rehausseur de tension

Pour la conversion 5V → 3.3V, l'adaptation s'est faite par un simple pont de résistance. (cf. Figure 11-a)

Et pour la conversion 3.3V → 5V , initialement, l'idée d'un montage rehausseur de tension (cf. Figure 11-b) utilisant une diode et une résistance de tirage avait été choisie et même malheureusement validée jusqu'au typon.
Le principe de ce montage est le suivant : On choisit une diode dont la chute de tension vaut 1V. Cette diode sera toujours passante, ainsi, lorsque A = 3.3V, B vaut 4.3V, et lorsque A = 0V, B vaut 1V

En fait, il se trouve que ce montage risque de ne pas marcher aux vues des niveaux de tension des PIC et du Xbee. En regardant dans les datasheet du Xbee et du PIC, on peut retrouver les valeur limites des états hauts et bas. Voici ces valeurs (en Volts) : pour le Xbee : $V_{OH} = V_{cc} - 0,5$; $V_{OL} = 0,5$; Pour le PIC : $V_{IH} = 0,8.V_{DD}$; $V_{IL} = 0,2.V_{DD}$

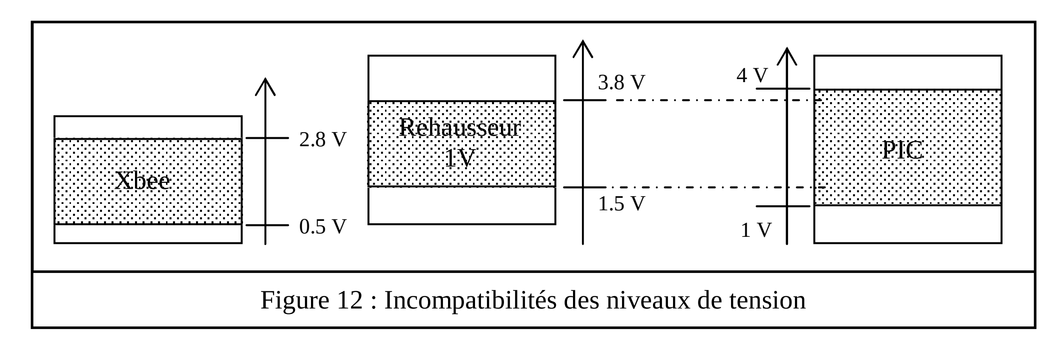


Figure 12 : Incompatibilités des niveaux de tension

Si un niveau haut émis par le Xbee est trop faible (2,8V), celui-ci sera rehaussé à 3,8V mais ne pourra pas être reconnu comme un état particulier par le PIC qui a une plage inconnue de 1V à 4V (lorsqu'il est alimenté à 5.0V). Par ailleurs, même l'état bas ne sera pas reconnu car il est trop haut.

À cause de ce désagrément, beaucoup de temps a été perdu. Une porte ET (Référence SN74AHCT1G08) a donc été soudée in extremis sur la face du dessous du circuit. En effet une porte ET est un circuit qui admet un niveau limite pour une tension haute en entrée égale à $V_{IH} = 2.8V$. En sortie, la porte ET retourne un état haut à 5V compatible avec le niveau du PIC.

Voici une photo montrant le soudage "in extremis" de la porte ET :

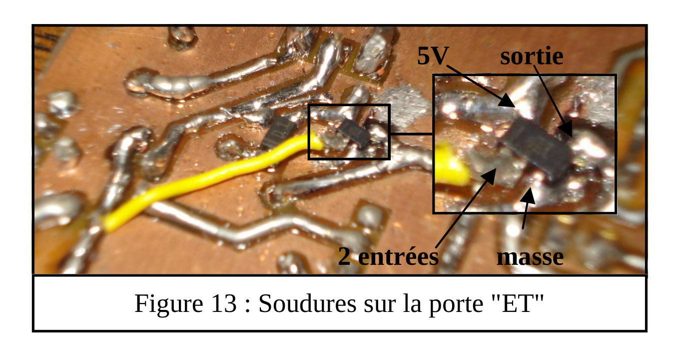


Figure 13 : Soudures sur la porte "ET"

Au final, le Xbee est câblé comme suit :

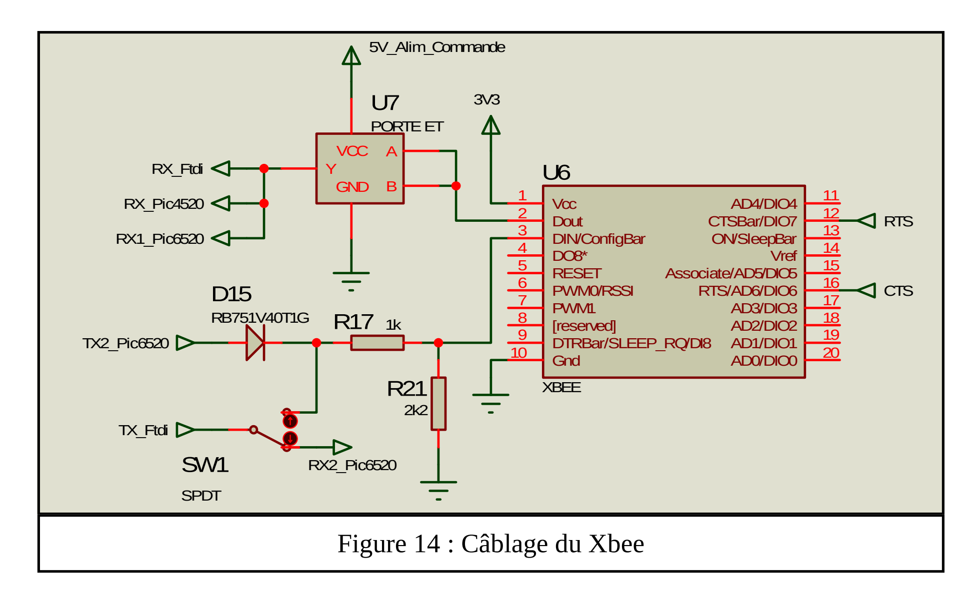


Figure 14 : Câblage du Xbee

La diode D14 sert à éviter que lors de la reprogrammation du Xbee, un signal puisse être envoyé sur le PIC 18F6520 (il ne s'agit que d'une sécurité). R18 et R19 forment le pont diviseur de tension 5V → 3.3V. La porte ET se charge donc de rehausser la tension 3.3V → 5V. Et enfin, pour permettre de le reprogrammer sans avoir à débrancher le Xbee, un interrupteur a été ajouté. Cet interrupteur déconnecte le 18F6520 de l'USB PC, pour permettre la communication bidirectionnelle entre le PC (USB) et le Xbee (sans cet interrupteur, on n'aurait que le sens Xbee→PC).

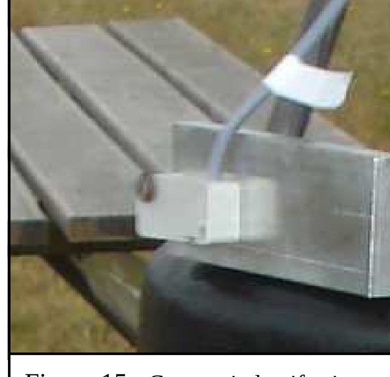
2) c - Câblage des PICs

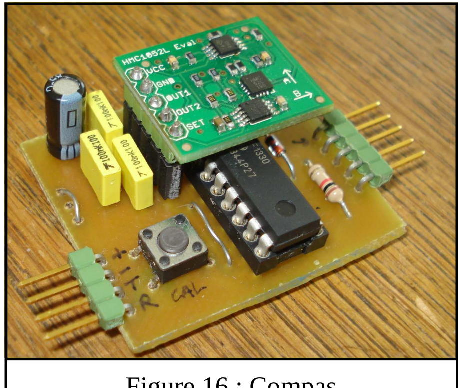
Les PICs sont cadencés à 12 MHz par un quartz extérieur. En plus de leurs connectiques essentielles (capteurs, commande des puces, etc…), ils sont tous les deux équipés d'un jeu de 6 LEDs de débogage (4 verte et 2 rouge). Le connecteur faisant office de programmateur PicKit2 peut également servir de débogage par UART : seul un interrupteur devra être basculé (cf. Figure 19). Les deux PICs sont découplés par un condensateur CMS de 100nF placés sur le typon au plus proche des PICs.

Les ponts diviseurs formés des résistances R13 à R16 permettent d'adapter le signal 12V issu des capteurs de inductifs montés sur l'antenne (cf. Figure 15)

La résistance R12 est bien entendu une résistance de tirage évitant un court circuit de l'alimentation lors de l'appui sur le bouton. Ce bouton est le bouton "Init" d'initialisation de la station sol.

Lors du développement de la carte, une idée a été émise concernant une initialisation complètement automatisée de la station grâce à un compas (une boussole). Cette boussole (cf. Figure 16) a été développée par Emmanuel Roussel. Pour plus de détails concernant une implémentation de celle-ci, il faudrait voir le rapport la concernant.


Figure 15 : Capteur inductif azimut


Figure 16 : Compas

Voici le schéma de câblage du PIC de commande des moteurs :

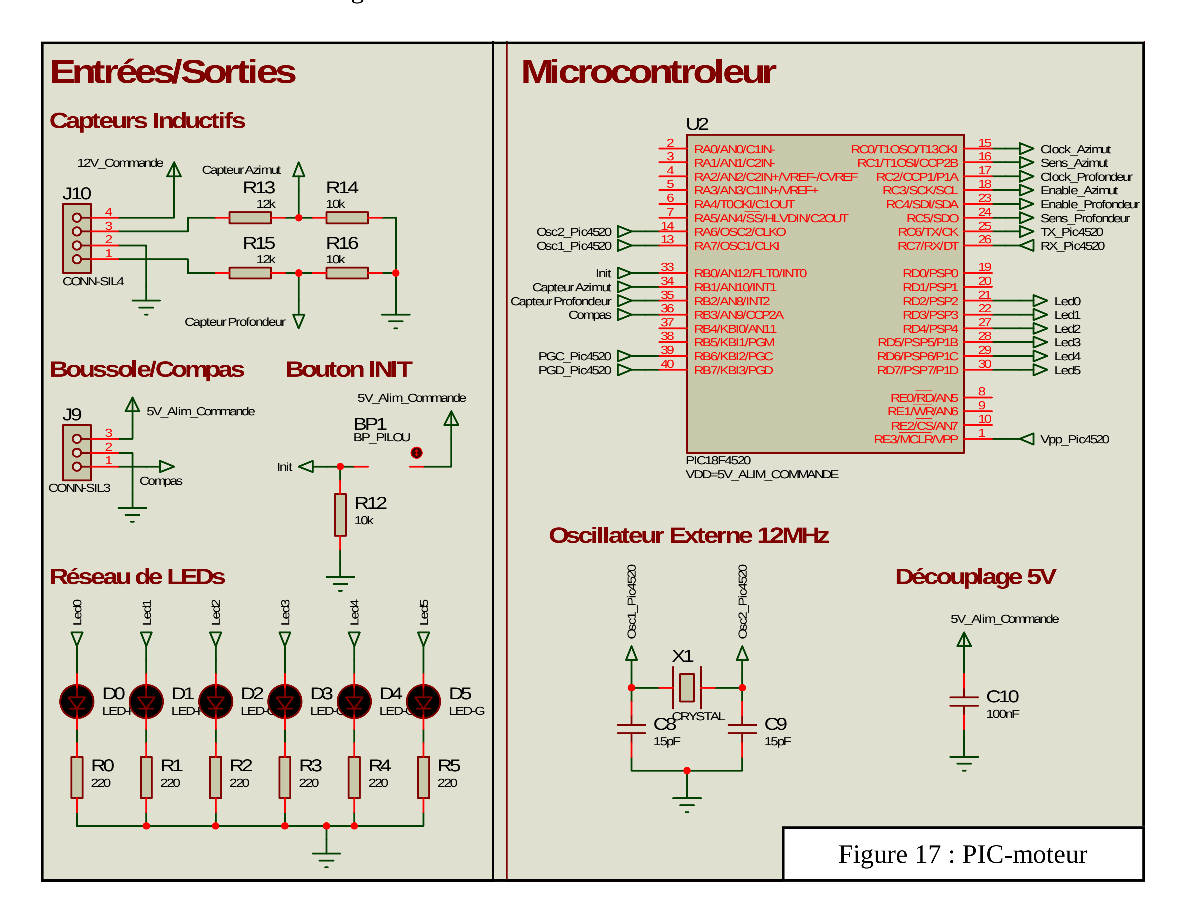


Figure 17 : PIC-moteur

Concernant le PIC d'aiguillage, le joystick est branché directement sur les pins CCP (Capture Compare). Ces pins permettent d'analyser facilement des signaux PWM.

Voici le schéma de câblage du PIC d'aiguillage :

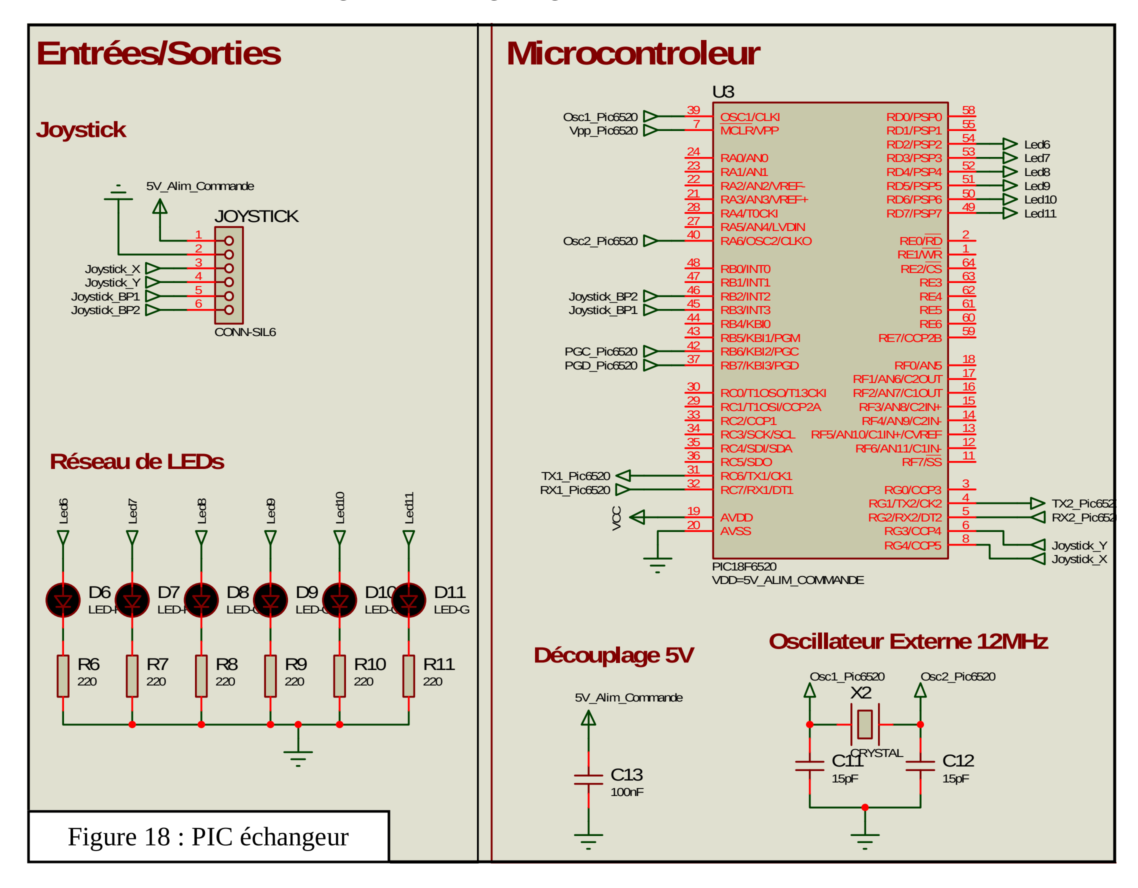


Figure 18 : PIC échangeur

Voici le schéma des connecteurs de programmation intégrant un port UART :

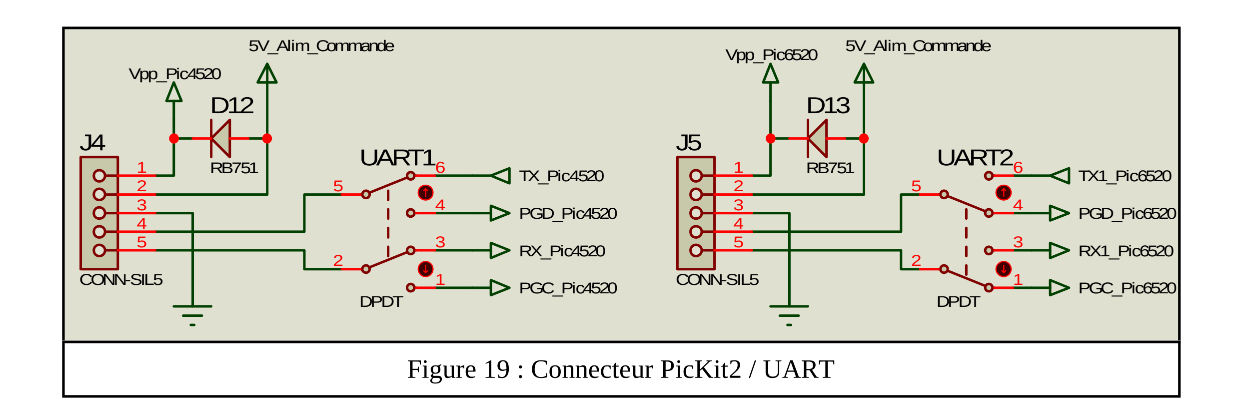


Figure 19 : Connecteur PicKit2 / UART

Les diodes D12 et D13 sont souvent utilisées sur ces connecteurs pour imposer un état haut sur la patte du MCLR tout en évitant un problème de niveau de tension (entre le 5V régulé et le 5V issu de l'USB PC) lors de la programmation. Il n'y a pas de critères spéciaux pour le choix de ces diodes : il faut juste avoir une diode CMS pas chère de tension de seuil inférieure à 1V (pour que la pin MCLR puisse lire un état haut : 4V est le minimum possible).

2) d - Connecteur RJ11 :

Sur la carte commande de la station sol V1, le connecteur avait mal été câblé puisque pour faire le lien avec le module de surimpression vidéo, il fallait croiser le câble RJ11 ce qui n'est pas très pratique. Pour cette raison, la nouvelle carte intègre un connecteur RJ11 qui lui est cette fois adapté au module de surimpression. Un câble droit devra donc être fait.

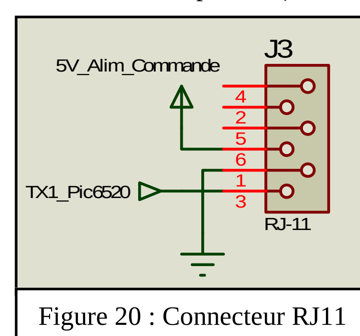


Figure 20 : Connecteur RJ11

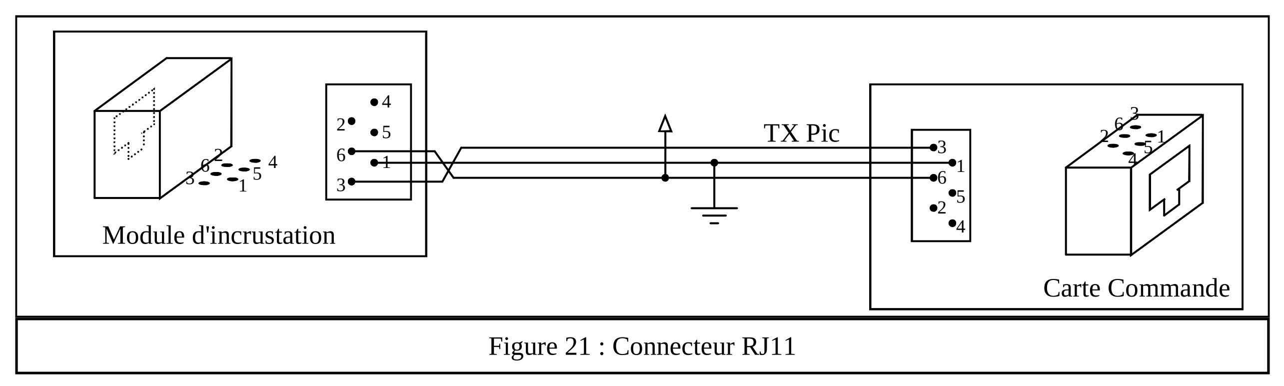


Figure 21 : Connecteur RJ11

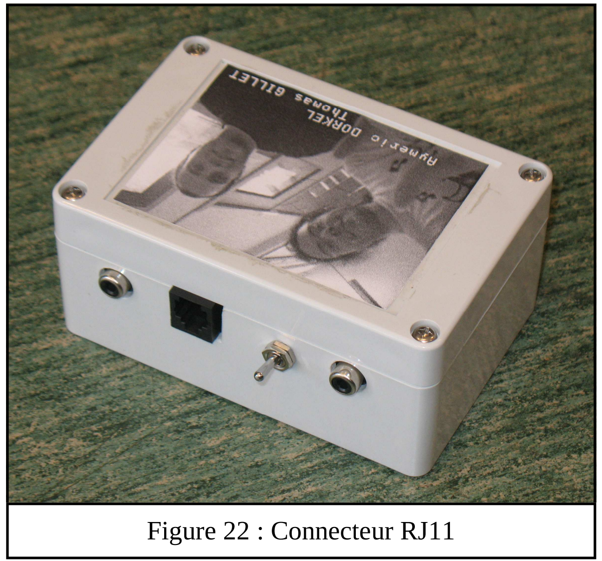


Figure 22 : Connecteur RJ11

2) e - Connecteur DIL10 Carte Commande/Puissance :

Contrairement à la station sol V1, le connecteur intègre ici une alimentation 12V issue de la carte puissance. Précédemment, cette alimentation était fournie par un deuxième câble d'alimentation directement depuis la batterie. L'alimentation 5V a aussi été intégrée à ce connecteur, ce qui permet de n'utiliser plus qu'un seul régulateur 5V qui se trouve sur la carte puissance. En revanche, les signaux "*Reset*" des deux puces L297 ont été retirés car ils ne sont pas utiles dans notre cas (cf. la suite de cette note d'application). Il s'agit en effet de signaux permettant de réinitialiser l'état initial des bobines à commander. Toute la commande se faisant de façon transparente pour nous, ces signaux ne nous sont pas vraiment nécessaires.

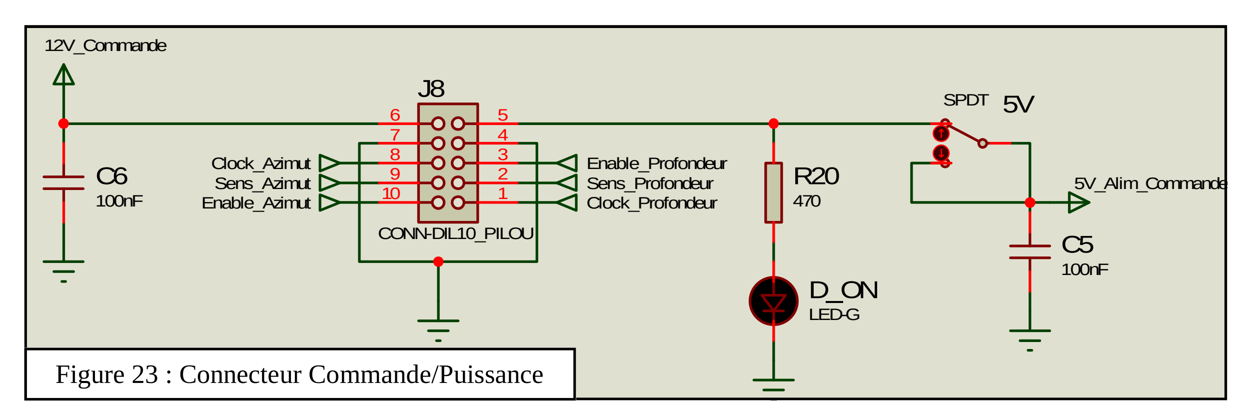


Figure 23 : Connecteur Commande/Puissance

Un interrupteur permet d'alimenter le 5V de la carte commande soit par le 5V régulé de la carte puissance, soit par n'importe quelle broche 5V de la carte commande (exemple PicKit etc…) . Une LED permet de visualiser si la carte puissance est alimentée ou non. Sur les schémas de "correction", une deuxième LED a été rajoutée pour indiquant l'alimentation de la carte commande. L'alimentation 12V est découplée par un condensateur CMS de 100nF en sortie de connecteur, et pour le 5V, il est régulé en sortie d'interrupteur.

2) f - Gestion de l'alimentation :

On utilise directement une tension 12V issue d'une batterie qui une fois filtrée alimente directement les hacheurs des moteurs. Une diode a été ajoutée en série pour éviter une inversion dans le sens des connecteurs. Pour former une tension 5V, un simple régulateur 5V a été choisi. Il permet d'alimenter les PICs de la carte commande, ainsi que les puces L297 et L298. Plus tard, la question du dimensionnement du régulateur a été abordée. Le régulateur 5V alimente au maximum 3 pics (les 2 principaux ainsi que celui du Joystick), 2 composants L297 et 2 L298.

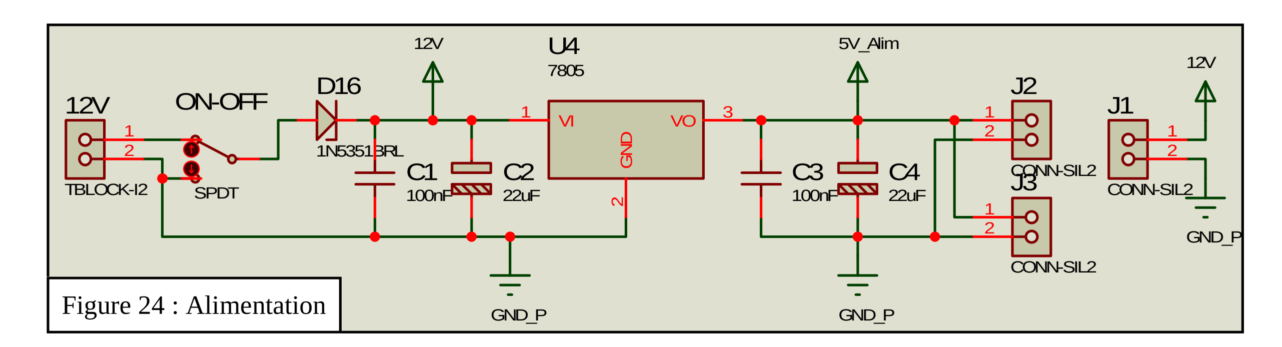


Figure 24 : Alimentation

Concernant les PICs, ils sont limités à 250mA chacun (cf. page 321 et 307 du 18F4520 et 18F6520), les L298 consomment au repos 36mA maximum chacun (cf. page 3 de la datasheet), et enfin, pour les L297, aucune indication n'est donnée. La détermination précise de la valeur maximale du courant que le régulateur devra fournir est difficile. Cependant, cette valeur est déjà majorée si on prend effectivement 250mA pour les PICs. En effet, ceux-ci ne consommeront pas autant puisque tous leurs périphériques ne sont pas utilisés. Par conséquent, on peut retenir pour valeur haute une consommation de 900mA.

Or les simples régulateurs 7805 permettent de délivrer un courant de 1A (voire plus avec un dissipateur adapté). De toutes façons, vu le différentiel de tension 12V → 5V, un dissipateur devrait être ajouté à ce composant. Pour informations, si jamais le choix de ce régulateur vient à être remis en question, le remplacement par un LM323T (dans le même package TO-220) pourra être envisagé.

Un dissipateur pour ce composant devra dissiper ($U_{batterie}$ - $U_{régulée}$) * $I_{consommée}$ = (12-5)*1 = 7 Watt.
Si l'on considère une température ambiante de 25°C, et que l'on souhaite obtenir une température de jonction de 30°C, on a : $\Delta T = P*R_{dissipateur}$ soit $R_{dissipateur}$ = 5°C/7W

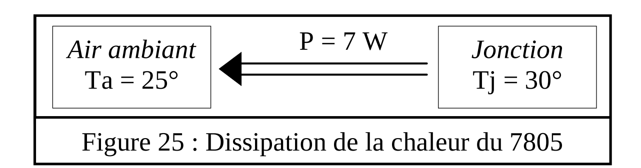


Figure 25 : Dissipation de la chaleur du 7805

Il faudra donc un dissipateur de résistance thermique au maximum de 0,7 °C/W (une valeur basse est préférable). Lors de hautes températures (ex à 40°C), la jonction sera à environ 35°C.

Concernant l'alimentation, trois connecteurs supplémentaires (2x5V et 1x12V) ont été ajoutés sur la carte puissance dans le cas où un accès à une de ces tensions serait nécessaire. Sur la carte commande, il y a également des points test (masse;tension) permettant d'accéder aux tensions 12V, 5V et 3.3V.

2) g - Commande des moteurs :

Nous avons 2 moteurs pas à pas à contrôler. Un moteur pas à pas est un ensemble de bobinages que l'on peut alimenter indépendamment les uns des autres. En alimentant certaines bobines du stator, on peut donc orienter le rotor (aimant permanent) selon des directions privilégiées. Selon le nombre de phases du moteur et le nombre de paires de pôles, un moteur pas à pas possède un certain nombre de positions privilégiées. En alimentant donc certaines bobines les unes après les autres, on peut donc faire avancer le moteur de pas en pas (cf. Figure 26).

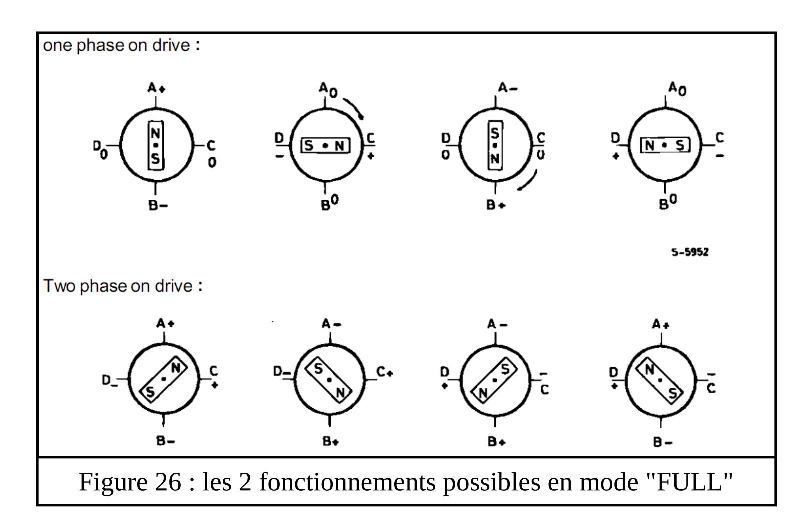


Figure 26 : les 2 fonctionnements possibles en mode "FULL"

Pour alimenter un moteur on utilise très souvent des hacheurs intégrés de type L298. Ici, il s'agit d'un moteur pas à pas. La commande n'est pas forcément évidente. Il existe des composants L297 qui remplissent la fonction de driver de moteur pas à pas.

Le schéma en Figure 27 est issu de la datasheet du L297 qui propose un schéma tout fait de commande de moteur pas à pas.

À ce schéma de base, certains ajouts ont été faits : un potentiomètre de contrôle du courant admissible a été ajouté (cf. Figure 28) ainsi qu'un interrupteur permettant de sélectionner le mode de commande du hacheur L298 : INH1, INH2 ou bien A,B,C,D. Ces deux modes de commande sont expliqués dans la datasheet.

Au final, il se trouve qu'il n'y a pas de grande différence entre ces deux modes. On pourrait donc à l'avenir penser à tirer la patte 11 du L297 à 5V.

Le filtre RC est un mécanisme de protection qui contrôle le courant des moteurs en faisant varier le rapport cyclique du hacheur. Ce n'est pas la même chose que l'entrée Clock qui elle contrôle la vitesse du moteur : à chaque front montant du signal Clock, l'alimentation des phases du moteur change (cf. Figure 26).

La pate 19 est à la masse, et donc la puce va fonctionner en mode "FULL", c'est-à-dire qu'il y aura 4 états successives des sorties A,B,C,D. Selon l'état initial du "translator" (composant principal du L297), on aura soit le fonctionnement "one-phase-on" soit le fonctionnement "two-phase-on" (cf. Figure 26).

Les résistances R22 et R23 sont des résistances "SENSE" de mesure du courant entrant dans les moteurs. Les puces L297 sont équipées de comparateurs de tension entre la tension aux bornes de ces résistances U_{sense} et la tension V_{ref} appliquée sur la patte 15 du circuit. Si la tension U_{sense} dépasse V_{ref} , alors, la puce modifie le rapport cyclique pendant un certain temps de façon à faire chuter la tension et donc le courant tiré par les moteurs.

Les résistances "SENSE" se fixent de manière à avoir une tension comprise entre 0 et 5V. Avec la formule $U_{sense} = R_s*I_{adm}$, et avec $R_s = 1\Omega$, on aura donc une tension de $U_{sense} = 2V$ lorsque le moteur consomme 2A. Pour pouvoir contrôler le courant admissible on place le potentiomètre RV (cf. Figure 28) sur l'entrée Vref. Pour limiter à 2A la consommation, il faudra donc tourner le potentiomètre jusqu'à obtenir une tension de 2V sur la patte 15 du L297. Il faudra bien veiller au calibrage de ces potentiomètre car si ils sont réglés à une tension maximale pour V_{ref}, alors, les moteurs pourront tirer 10A, ce qui risque d'être assez dangereux.

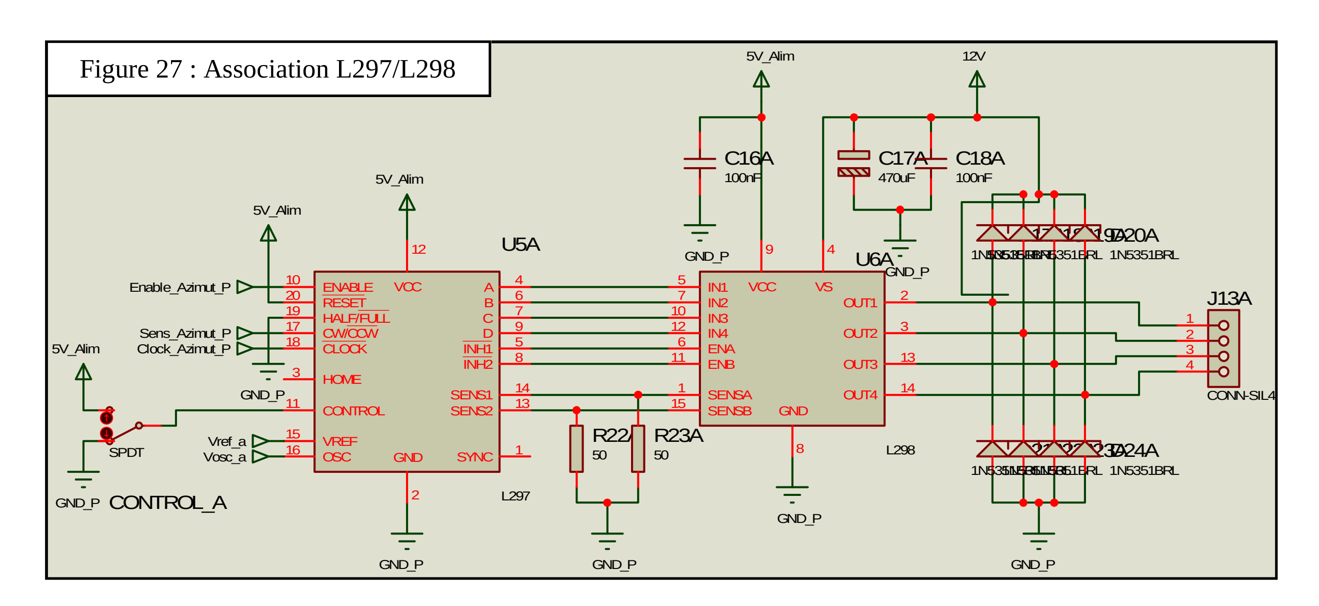


Figure 27 : Association L297/L298

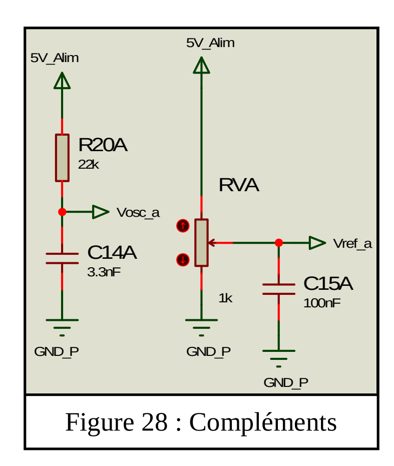


Figure 28 : Compléments

3) Réalisation des cartes :

Afin de gagner en place, la station sol sera réalisée comme précédemment sur deux cartes "Commande" et "Puissance". Deux circuits intégrés ont été choisis en CMS car il n'y avait pas le choix (le Ftdi ainsi que le PIC 18F6520). D'autres composants plus classiques (résistances, diodes, condensateurs, LEDs) ont également été choisis en CMS.

Le PIC "moteur" 18F4520 aurait pu être choisi en CMS. Cependant, la carte étant nouvelle (prototype et non une version finale), et le tirage de celle-ci n'étant pas professionnel, le format DIL standard a été préféré à celui "CMS". Sur les cartes présentées ci après, les plans de masse ont été retirés.

3) a - Typon carte commande :

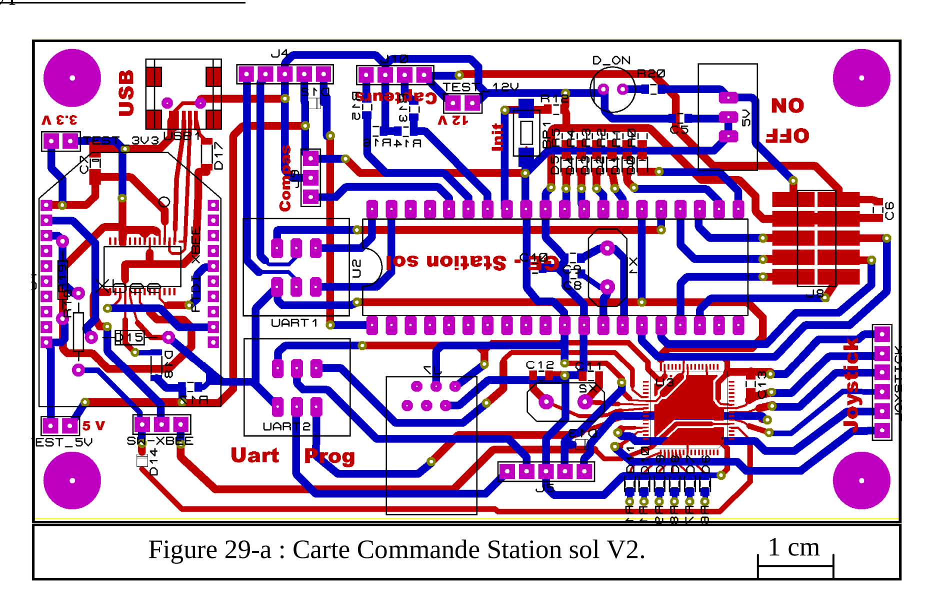


Figure 29-a : Carte Commande Station sol V2. 1 cm

Le typon présenté est le typon d'origine. Après débogage de la carte, le typon a été modifié. En effet, voici un listing des défaut relevés sur ce premier typon :

- Problème avec le montage rehausseur de tension (Diode+Résistance), remplacé par une porte ET.
- Le package du connecteur DIL10 n'est pas bon si on utilise un plan de masse (toutes les pattes étaient reliées). Remplacement de ce package par le package d'origine en utilisant l'option "Change Layer → Solder Side"
- Le connecteur RJ11est mal situé car lorsque l'on superpose cette carte avec la carte puissance, ce connecteur est mal situé (au niveau du radiateur du hacheur du moteur de profondeur).
- Une deuxième LED d'état d'alimentation de la carte commande a été ajoutée.

Voici donc le typon une fois les défauts précédemment cités corrigés :

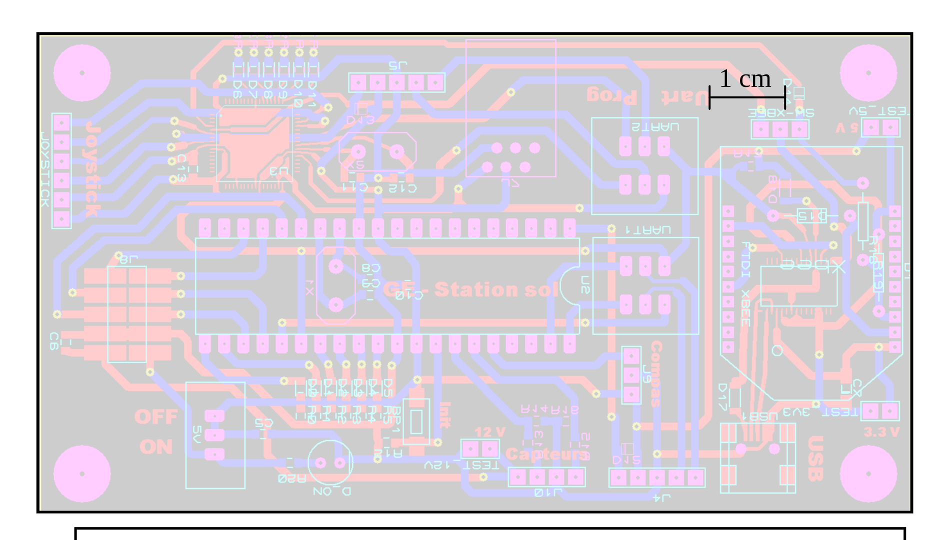


Figure 29-b : Carte Commande Station sol V2. Typon Corrigé

3) b - Typon carte puissance :

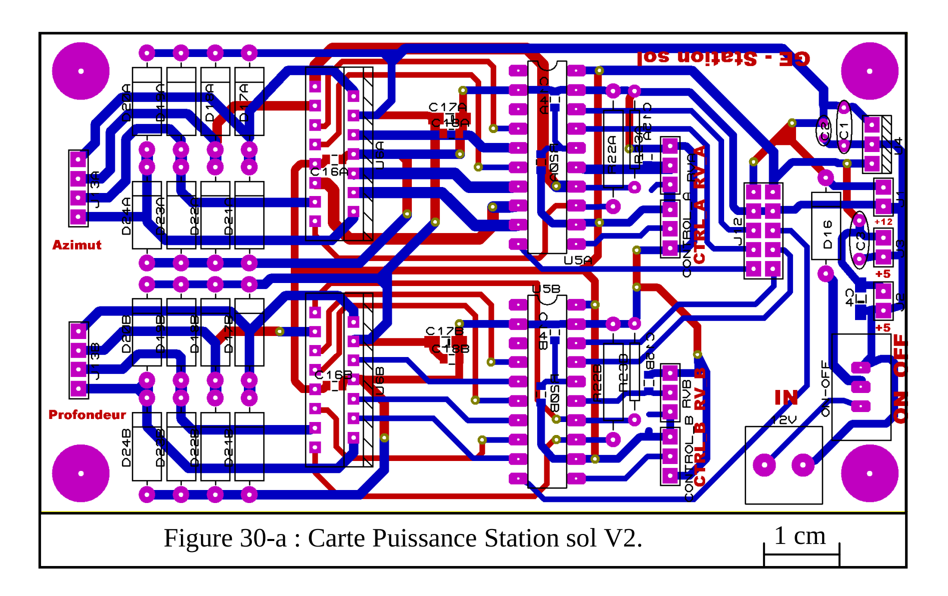


Figure 30-a : Carte Puissance Station sol V2. 1 cm

Cette carte possède elle aussi quelques défauts :

- Package des Résistances "SENSE" R22 et R23.
- Le régulateur est trop proche des connecteurs autour. Il faudra prévoir un emplacement pour un dissipateur.
- Vérifier le package des condensateurs CMS
- Il faut éloigner les potentiomètres à la fois du switch "CONTROL" ainsi que des résistances 22 et 23. Il faudrait également tourner de 180° un de ces potentiomètres afin que le réglage du courant admissible se fasse dans le même sens pour les deux étages moteurs. (Par exemple si on visse, on augmente le courant limite, si on dévisse, on diminue cette limite). Pour plus de clarté, on pourrait faire apparaitre dur le package la vis de réglage.
- Le changement du bornier d'alimentation générale 12V pourrait être envisagé. On pourra choisir une solution à clips, plus pratique pour le montage/démontage.
- Pourquoi pas supprimer les interrupteurs "CTRL_x" car comme dit en page 13, il ne sont pas utiles.

Voici donc le typon une fois les défauts précédemment cités corrigés :

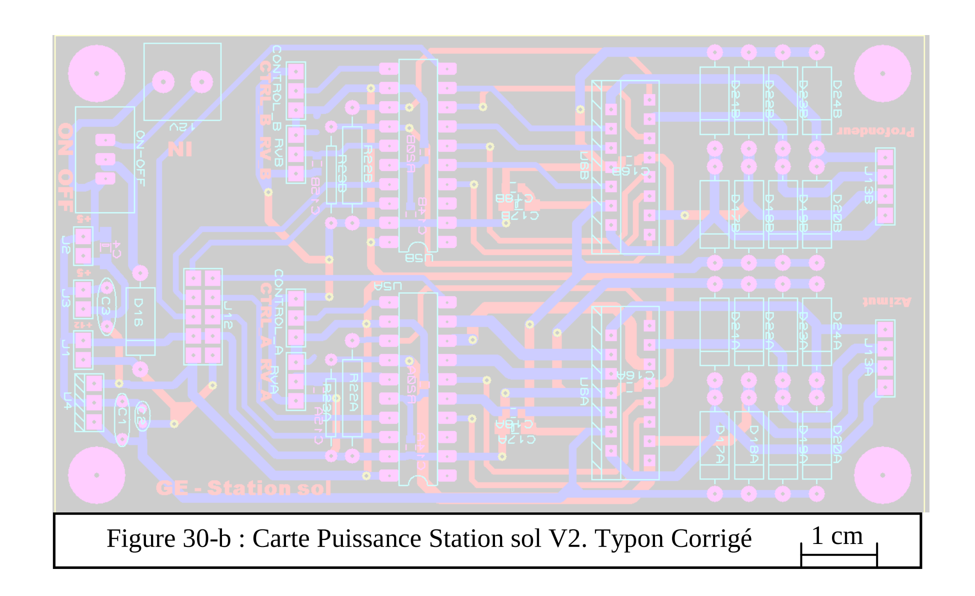


Figure 30-b : Carte Puissance Station sol V2. Typon Corrigé 1 cm

3) c - Prototype de la station :

Voici une image présentant les emplacements des différents composants/connecteurs. On remarquera qu'un soin particulier a été apporté à la logique de placement de ceux-ci.

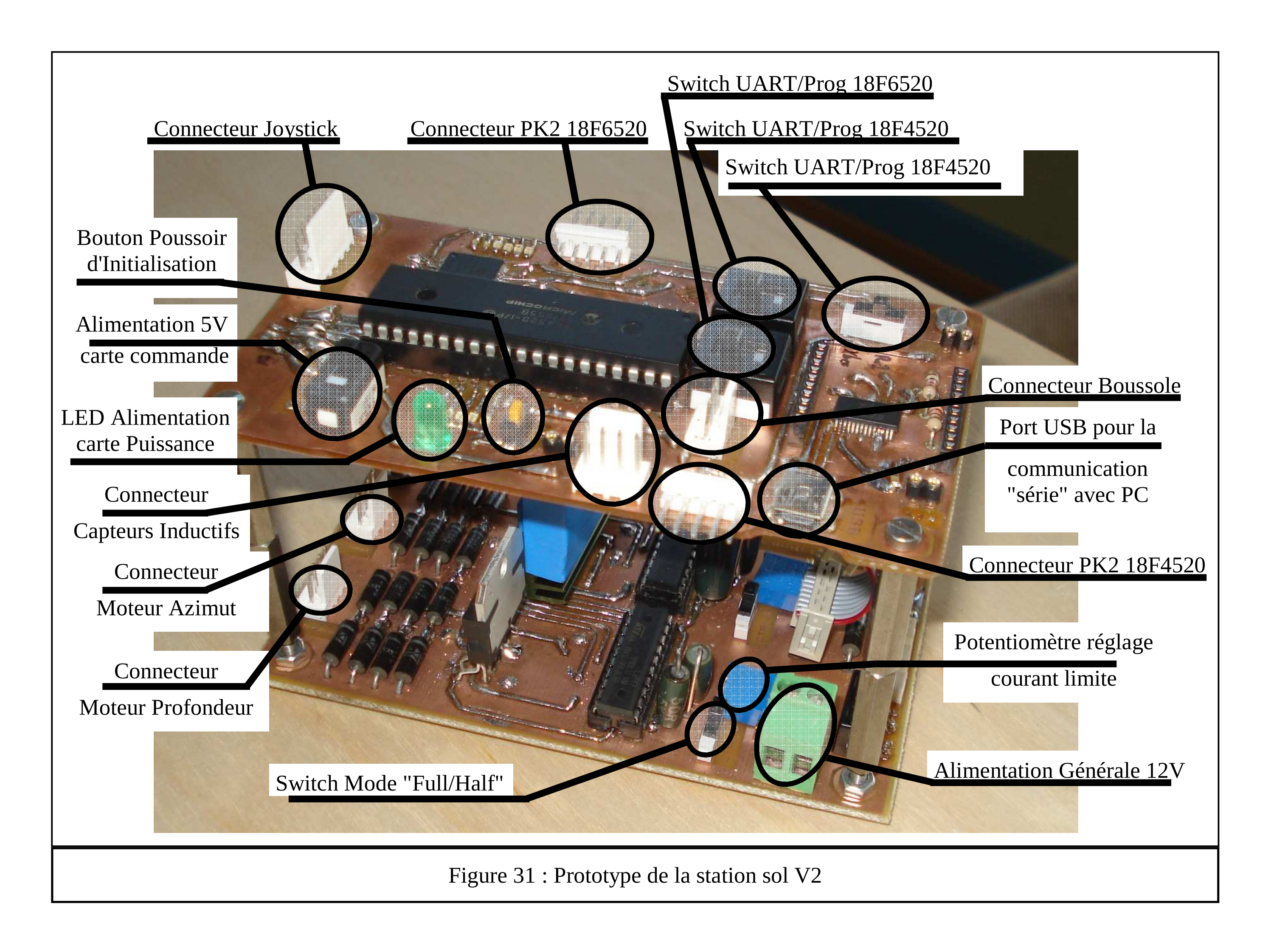


Figure 31 : Prototype de la station sol V2

3) d - Tests effectués sur la carte puissance :

Afin d'effectuer une première vérification du fonctionnement de la carte puissance, un signal Clock issu d'un GBF a été imposé sur le connecteur DIL10 et une mesure des signaux de sortie ont été effectués. Lors de ces tests, les moteurs ont bien fonctionné comme il le fallait. Voici sur les figures 19 et 20, les tensions relevées sur l'entrée Clock de la puce L297 ainsi que sur les bobines A, B, C, D du moteur

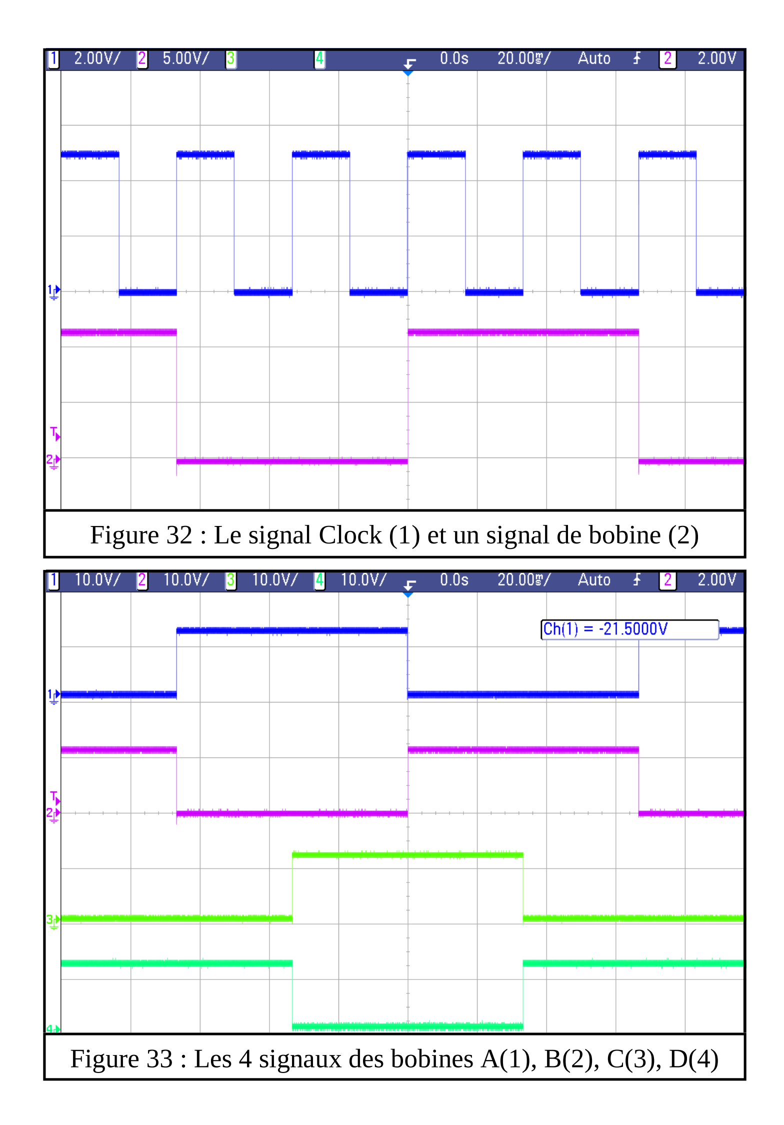


Figure 32 : Le signal Clock (1) et un signal de bobine (2)

Figure 33 : Les 4 signaux des bobines A(1), B(2), C(3), D(4)

3) e - Commande des composants :

La commande a été effectuée sur le site de Farnell. Voici la liste de composants nécessaires pour ces circuits :

1 x PIC 18F4520
1 x PIC 18F6520
1 x Ftdi FT232RL
1 x Xbee Pro
1 x 7805
2 x L297
2 x L298
1 x Porte ET Cms
2 x Quartz HS 12MHz

4 x Condensateurs Cms 15pF
12 x Condensateurs Cms 100nF
1 x Condensateur Plastique 100nF
2 x Condensateur Chimique 22µF
2 x 470 uF
3.3nF

8 x LEDs Cms Vertes
4 x LEDs Cms Rouges
2 x LEDs Simple Vertes

12 x Résistances Cms 220Ω
3 x Résistances Cms 10kΩ
2 x Résistances Cms 12kΩ
1 x Résistance Cms 1kΩ
1 x Résistance Cms 2.2kΩ
2 x Résistance Cms 22kΩ
4 x Résistance de Puissance 0.5Ω
2 x Résistances LEDS xxxxxxxxxx
2 x Potentiomètre 10kΩ

5 x Diodes (chute de tension faible)
17 x Diodes (Roue Libre et Alim)

2 x Connecteurs SIL5 (pas 2.54mm)
1 x Connecteur SIL3 (pas 2.54mm)
3 x Connecteur SIL4 (pas 2.54mm)
1 x Connecteur SIL6 (pas 2.54mm)
2 x Connecteurs SIL2 (pas 2.54mm)
2 x Connecteurs DIL10 (pas 2.54mm)
2 x Connecteurs Femelle DIL10 (pas 2.54mm)
1 x Connecteur RJ11
1 x Bornier d'Alimentation Générale
3 x Barrettes sécables SIL2 (pas 2.54mm)
2 x Barrettes sécables SIL10 (pas 2.00mm)
1 x Port USB
2 x Petits Interrupteurs SPDT
2 x Interrupteurs SPDT
2 x Interrupteurs DPDT
1 x BP Cms

III - Descriptif du programme :

Il y a deux programmes sur cette station sol car deux microcontrôleurs. Le microcontrôleur responsable de la commande des moteurs est le 18F4520, tandis que celui qui fait l'aiguillage des données est le 18F6520. Pour le moment, seul le programme des moteurs a été fait.

1) PIC motorisation :

Le but de ce code est de prendre en compte la position du drone (coordonnées GPS et altitude) et après une phase d'initialisation (coordonnées de référence de la station sol), les moteurs pas à pas sont commandés en nombre de pas à effectuer pour s'incliner en direction du drone sachant qu'un tour d'azimut vaut 488 pas et un tour profondeur vaut 246 pas.

1) a - Organigramme :

Voici un organigramme très simplifié du code :

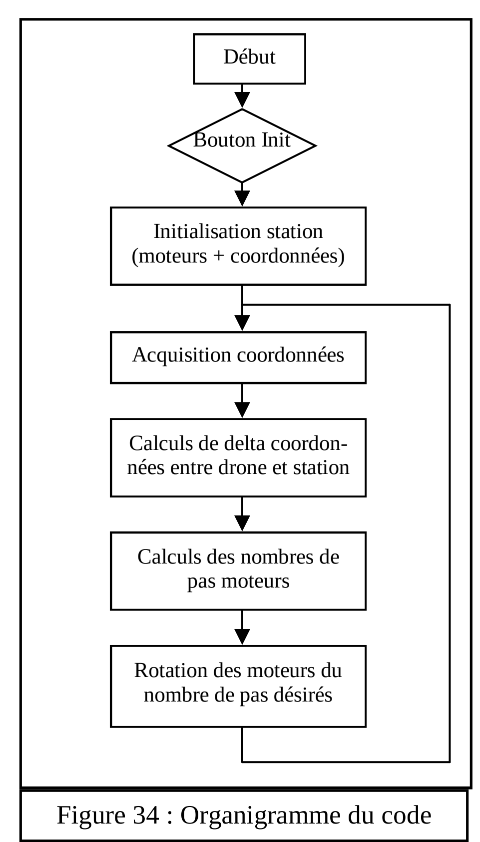


Figure 34 : Organigramme du code

1) b - Fonctionnement du code :

Actuellement, la réception de la trame se fait sous interruptions : dès que l'on active le bit d'activation des interruptions réception UART (*RCSTAbits.CREN*), les valeurs de longitude, latitude et d'altitude du drone sont chargées dans les variables correspondantes (*longitude_GPS*, *latitude_GPS* et *altitude_GPS*).

Une fois ces valeurs chargées (ce qui se produit lorsque la fonction *acquisition_GPS()* est appelée), la position des deux moteurs étant connue (variables *position_moteur_azimut* et *position_moteur_profondeur*), le programme va calculer le nombre de pas à effectuer pour chacun des deux moteurs.

En premier lieu, le moteur azimut est commandé (*tracking_azimut(1)*) puis c'est au tour du moteur de profondeur (*tracking_profondeur(5)*). Dans le code des fonctions de "tracking", une horloge est créée. Cette horloge est nécessaire, et c'est elle qui va faire bouger le moteur d'un pas à l'autre à chaque front montant. À chaque cycle de cette horloge, les variables de position des moteurs sont incrémentées (ou décrémentées selon le sens de rotation).

Les étapes de suivi des deux moteurs sont actuellement séquentielles : on charge la trame, on calcule le nombre de pas on bouge les moteurs, puis on recommence. Le système fonctionne donc en saccadé. Il serait beaucoup plus intéressant de faire une lecture en continu de la trame, ainsi les moteurs tourneront en continu, ce qui permettra de rendre le système plus fluide. Cela dit, les moteurs sont assez rapides pour arriver à la position souhaitée très rapidement, et à l'échelle du déplacement du drone, les moteurs devraient fonctionner assez bien dans l'état actuel des choses. C'est d'ailleurs le cas actuel : lors du suivi, il n'y a pas de décrochage dû à une vitesse trop grande du drone.

1) c - Stratégie de calcul des pas moteurs :

Le calcul de l'angle du drone est donné par la simple formule suivante (cf. la figure :

$$angle = \arctan\left(\frac{longitude_GPS - longitude_station}{latitude_GPS - latitude_station}\right)$$

Une fois la valeur calculée, il faut également savoir dans quel cadran le drone se trouve. En effet, la fonction arctan() est une fonction de] -∞ ; +∞ [vers] -π/2 ; π/2 [. Sur la Figure 36, la valeur des angles calculée est donnée par *valeur* et la valeur de l'angle réelle est donnée par *angle*. Au final, la fonction utilisée retourne un angle compris dans l'intervalle] -π ; π [(sachant que l'azimut Nord correspond à l'angle 0).

Une fois cet angle obtenu, il reste à calculer le delta angle entre cet angle et celui où se trouve le moteur. L'angle du moteur est donnée par la formule :

$$angle_moteur_azimut = position_moteur_azimut . \frac{2.pi}{488}$$

Concernant la soustraction, il faut s'assurer que la différence est belle et bien inférieure à π puis savoir de quel signe est cette différence. Le signe sera donné par une variable spécifique (*sens_rotation_azimut*)

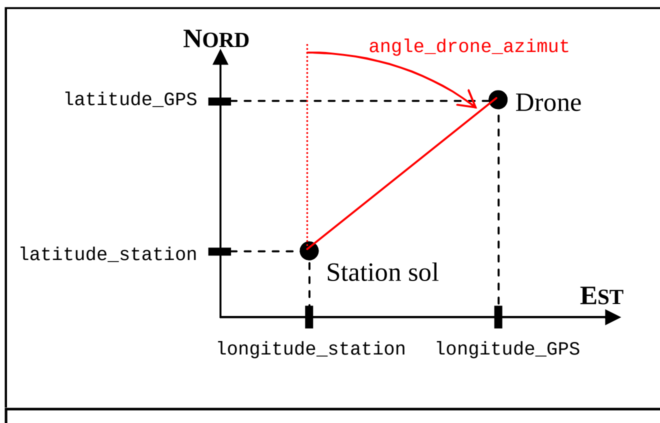


Figure 35 : Schéma pour le calcul de l'azimut

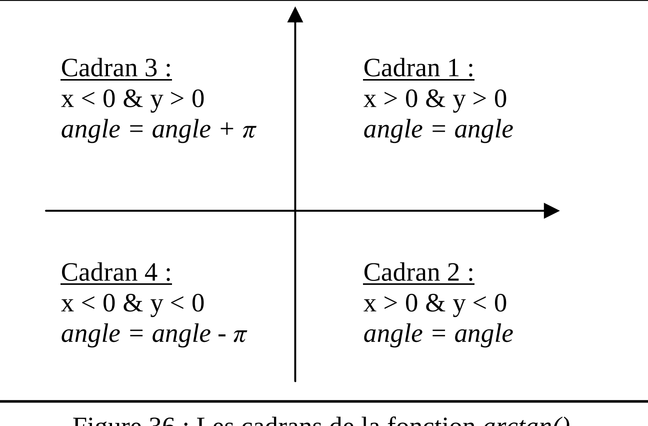


Figure 36 : Les cadrans de la fonction *arctan()*

Pour ce qui est du calcul de l'angle de la profondeur, il faut prendre en compte la distance de la projection au sol de la station sol et du drone (cf. la cotation "distance" sur la Figure 25). L'angle de profondeur sera finalement donné par la formule :

$$angle = \arctan\left(\frac{altitude_GPS - altitude_station}{distance}\right)$$

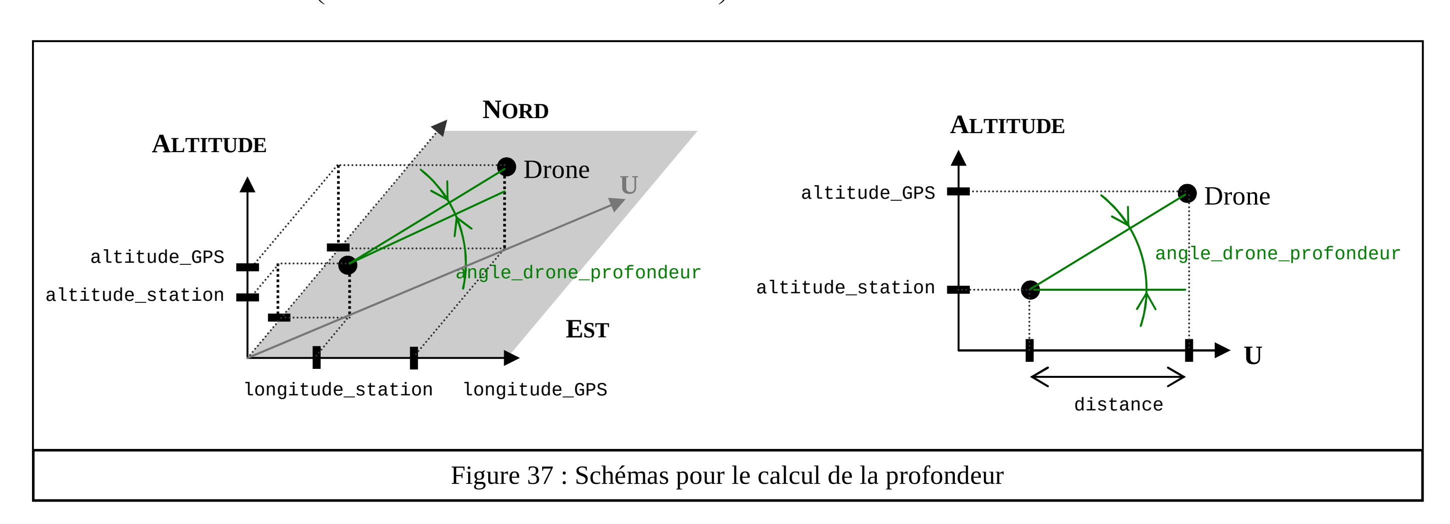


Figure 37 : Schémas pour le calcul de la profondeur

1) d - Détails du code :

Afin de permettre une plus grande rapidité de prise en main du code de la station à l'avenir, voici des extraits détaillés du code (certains passages ont été supprimés/modifiés pour plus de lisibilité) :

La fonction main :

```
void main(){

    /* déclaration de variables  */

    while(INIT == 0 ) ;
    initialisation_station() ;
    while(1) {
        acquisition_GPS() ;
        if ((latitude_GPS!=0)&&(longitude_GPS!=0)) break ;
    }
    longitude_station = longitude_GPS ;
    latitude_station = latitude_GPS ;
    altitude_station = altitude_GPS ;
```

→ **Attente d'appui sur le bouton pour initialiser la station (moteurs et coordonnées GPS)**

```
    while(1) {

        acquisition_GPS();
        coord_x = (long) longitude_GPS - (long) longitude_station ;
        coord_y = (long) latitude_GPS - (long) latitude_station ;
        coord_z = altitude_GPS - altitude_station ;
```

→ **Calcul du delta position entre la station et le drone**

```
        angle_drone_azimut = angle_azimut_du_drone(coord_x,coord_y) ;
        angle_moteur_azimut = position_moteur_azimut*2.0*pi/488.0 ;
        if( fabs(angle_moteur_azimut-angle_drone_azimut) > pi ) {
            delta_angle_azimut = 2*pi - fabs(angle_moteur_azimut-angle_drone_azimut) ;
            if (angle_moteur_azimut-angle_drone_azimut>0) sens_rotation_azimut = SENS_HOR ;
            else sens_rotation_azimut = SENS_TRIGO ;
        }
        else {
            delta_angle_azimut = fabs(angle_moteur_azimut-angle_drone_azimut) ;
            if (angle_moteur_azimut-angle_drone_azimut>0) sens_rotation_azimut = SENS_TRIGO ;
            else sens_rotation_azimut = SENS_HOR ;
        }
        nb_pas_azimut = (int) (0.5+(delta_angle_azimut*488.0/(2.0*pi))) ;
```

→ **Calcul du nombre de pas pour le moteur azimut**

```
        tracking_azimut(1) ;
```

→ **Déplacement du moteur azimut**

```
        x = (int) (11.11949266*coord_x) ; // en mètres
        y = (int) (11.11949266*coord_y) ; // 111194.92... =  Rayon_Terre * 2pi / 360
        angle_drone_profondeur = angle_profondeur_du_drone(x,y,coord_z) ;
        angle_moteur_profondeur = position_moteur_profondeur*2.0*pi/246.0 ;
        delta_angle_profondeur = fabs(angle_moteur_profondeur-angle_drone_profondeur) ;
        if ( angle_moteur_profondeur < angle_drone_profondeur ) sens_rotation_profondeur = SENS_HOR ;
        else sens_rotation_profondeur = SENS_TRIGO ;
        nb_pas_profondeur = (int) (0.5+(delta_angle_profondeur*246.0/(2.0*pi))) ;
```

→ **Calcul du nombre de pas pour le moteur profondeur**

```
        tracking_profondeur(5) ;
```

→ **Déplacement du moteur profondeur**

```
    }
}
```

La fonction de déplacement des moteurs :

```
void tracking_azimut(int vitesse) {
    int i = 0 ;
    Azimut_SensRotation_297 = sens_rotation_azimut ;
```

→ **Choix du sens de rotation du moteur (la variable sens_rotation_azimut est globale)**

```
    Azimut_Enable_297 = 1 ;
    for(i=0;i<nb_pas_azimut;i++) {
        Azimut_Clock_297 = 0;
        Delay10KTCYx(vitesse);
        Azimut_Clock_297 = 1;
        Delay10KTCYx(vitesse);
        if (sens_rotation_azimut==SENS_HOR) position_moteur_azimut ++ ;
        else if (sens_rotation_azimut==SENS_TRIGO) position_moteur_azimut -- ;
    }
}
```

→ **Génération d'un signal horloge pour le déplacement des moteurs et incrémentation de la variable de position du moteur.**

La gestion des interruptions :

```
void InterruptHandler(){

    if ((PIR1bits.RCIF)&&(MAJ_Trame==1)) {
        if(RCREG=='@') compteur_octet = -1 ;
```

On détecte @

```
        else if (RCREG=='#') {
            if (compteur_octet==-1) compteur_octet=0;
```

À ce stade là, on a détecté consécutivement @ et #
On pourra donc inscrire le reste de la trame dans le tableau trame[]

```
            else compteur_octet=-2;
```

Si # n'a pas été détecté, on attendra de recevoir à nouveau @

```
        }
        else if ((compteur_octet>=0)&&(compteur_octet<18)){
            trame[compteur_octet]=RCREG ;
            compteur_octet++;
            if(compteur_octet>=18) {
                MAJ_Trame = 2 ;
                compteur_octet = -2 ;
                RCSTAbits.CREN = 0;
            }
        }
```

Ici, on récupère la trame et on la stocke dans le tableau trame[].
Une fois que la trame est complètement chargée, on désactive les interruptions sur USART

```
    }
}
```

L'acquisition des données GPS :

```
void acquisition_GPS() {
        unsigned long degre ;
        unsigned long minute ;
        unsigned long datarecue ;

        MAJ_Trame = 1 ;
        RCSTAbits.CREN = 1;
        while(MAJ_Trame != 2) ;
```

Ici, on active les interruptions sur USART, et on attend ensuite que la trame soit chargée complètement

```
        datarecue = (((unsigned long)trame[0])<<24) + (((unsigned long)trame[1])<<16)
                 + (((unsigned long)trame[2])<<8) + (((unsigned long)trame[3])) ;
        degre = (datarecue/1000000)*1000000 ;
        longitude_GPS = (degre + ((datarecue - degre)/60)*100)/100 ;

        datarecue = (((unsigned long)trame[4])<<24) + (((unsigned long)trame[5])<<16)
                 + (((unsigned long)trame[6])<<8) + (((unsigned long)trame[7])) ;
        degre = (datarecue/1000000)*1000000 ;
        latitude_GPS = (degre + ((datarecue - degre)/60)*100)/100 ;
```

Ici, on traite les données de longitude/latitude pour la mettre en forme sous un nombre en degrés pur :
Ex: la trame reçue est sous la forme ddmmmm (d:degré,m:minute), et on la met sous forme dddddd (x10^{-4}).

```
        altitude_GPS = (trame[12]<<8)+trame[13] ;
```

L'altitude est déjà sous la bonne forme.

```
        MAJ_Trame = 0 ;
}
```

La conversion des delta position drone-station en angle (seul le code de l'azimut apparaît ici) :

```
float angle_azimut_du_drone(long coordonnee_x,long coordonnee_y) {
    float angle = 0 ;
    if(coordonnee_y==0) {
        if(coordonnee_x==0) {}
        else if(coordonnee_x>0) angle = pi/2.0 ;
        else if(coordonnee_x<0) angle = -pi/2.0 ;
    }
    else {
        if(coordonnee_x==0) {
            if(coordonnee_y>0) angle = 0 ;
            else angle = pi ;
        }
        else {
            angle = atan(((float)coordonnee_x)/((float)coordonnee_y)) ;
            if(coordonnee_y<0){
                if (coordonnee_x<0) angle = angle - pi ;
                else angle = angle + pi ;
            }
        }
    }
    return angle ;
}
```

On calcule la valeur de l'angle de]-pi;+pi°], tout en prenant en compte les différents cadrans de la fonction arctan()

2) PIC échangeur :

Voici un rappel du rôle de ce PIC. Tout d'abord, il réceptionne en continu la trame GPS. Il va ensuite récupérer par UART une trame PID depuis le PC (via un port USB). Il récupère également les consignes Joystick. Il envoie ensuite une trame unique contenant les PIDs ainsi que les consignes Joystick par Xbee au drone.

Par ailleurs, il fait la réception de la trame GPS pour pouvoir afficher (selon les consignes du PC) des informations sur la lunette 3D.

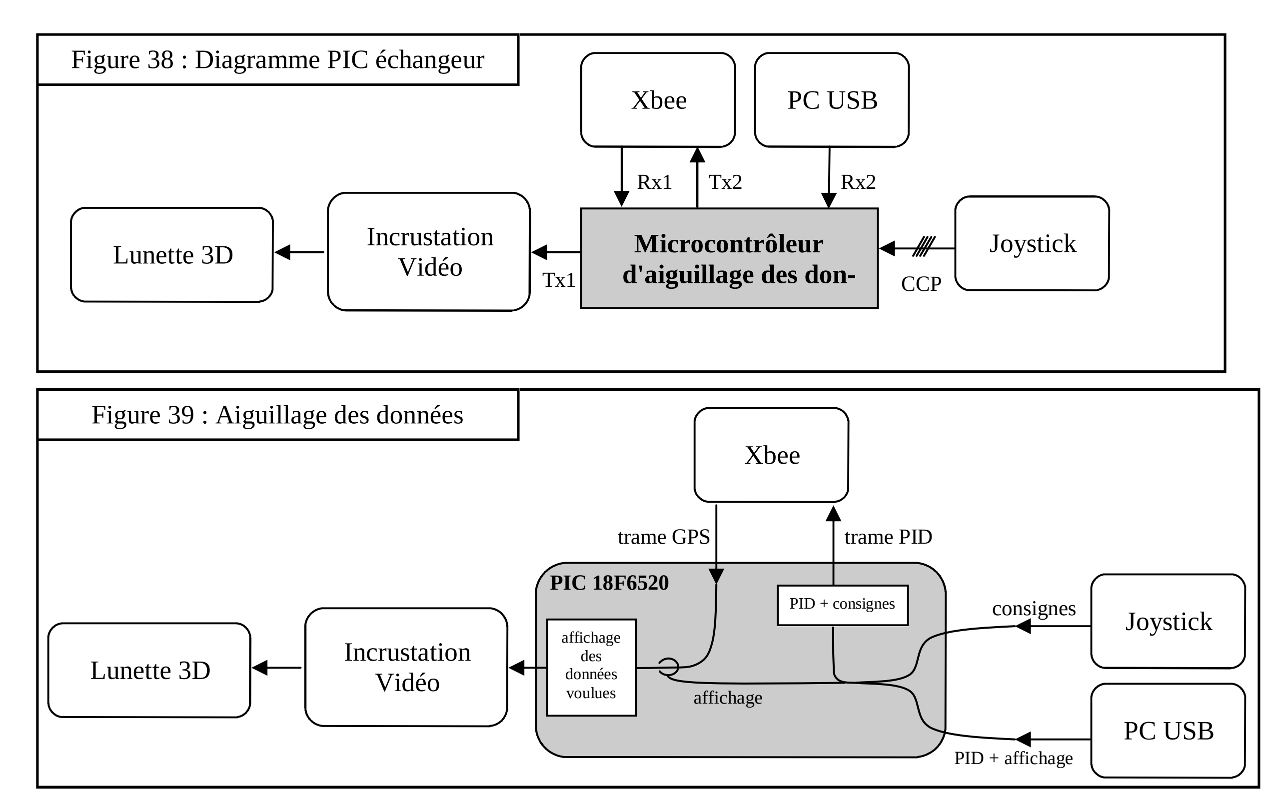


Figure 38 : Diagramme PIC échangeur

Figure 39 : Aiguillage des données

Idées pour le code :

Le code de ce PIC a été commencé partiellement par Emmanuel Roussel (GE4), des idées pour la personne qui s'en occupera sont également données à titre indicatif dans la suite. Elles ne sont peut être pas réalisables mais ça permettra de refléter ce qui a été pensé lors de la réalisation de la carte :

Pour le choix de l'affichage des données, au début (une fois la station sol démarrée), une trame PID issue du PC pourra par exemple envoyer un code bien précis indiquant si oui ou non telle information devra être affichée. Suite à cela, le PC n'aura plus besoin d'envoyer ce qu'il faut afficher. En interne, le PIC filtrera les données issue du Xbee à afficher sur la surimpression.

On pourrait également concaténer la consigne Joystick avec les corrections USB, ou bien créer deux trames différentes envoyées tour à tour. (Tout dépend de ce qui est préférable pour la carte mère embarquée).

IV - Mode d'emploi de la station sol V2 :

L'alimentation de la carte se fait par une batterie 12V qui peut fournir suffisamment de courant (des tests sur des alimentations stabilisées de labo n'ont pas été concluant puisque les moteurs consommaient beaucoup trop pour celles-ci).

Brancher le Xbee et vérifier la configuration du Xbee;

Au début, seule la LED 1 s'allumera.

Appuyer sur le bouton INIT.

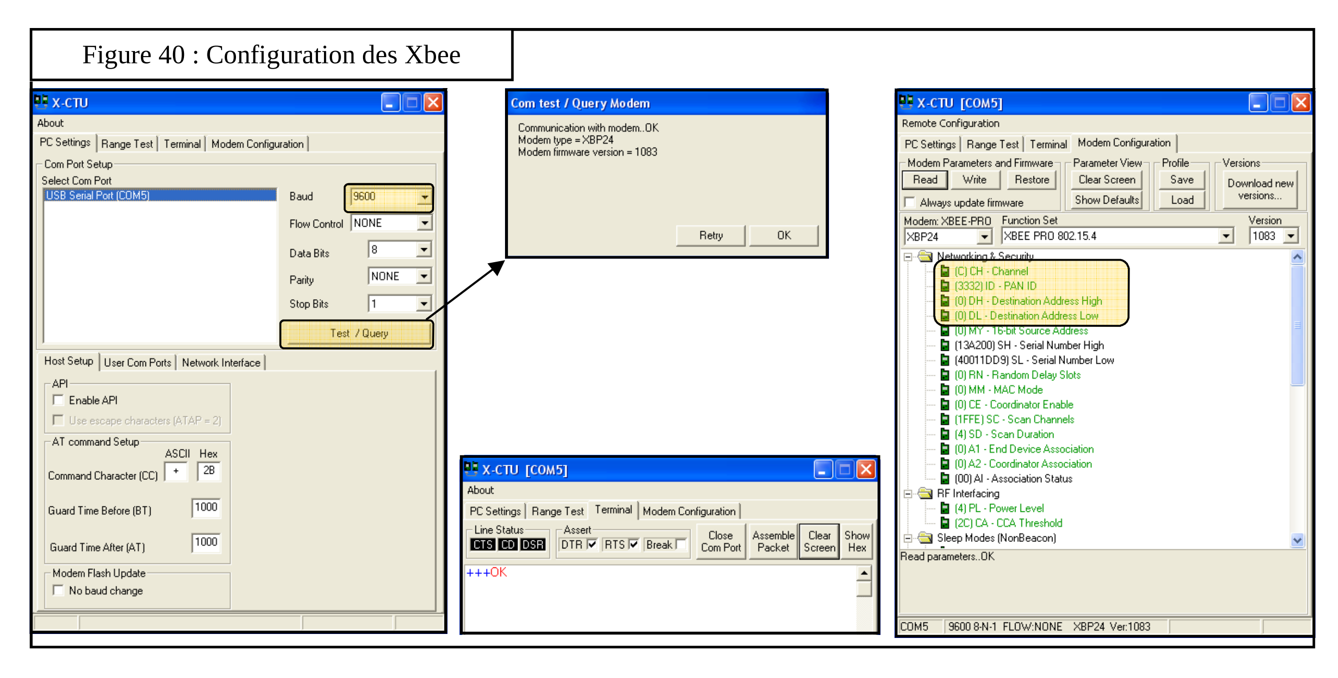
Une fois que le drone est démarré et que la carte capteur du drone envoie la trame GPS correctement, la LED 2 s'allumera.

Commence alors la routine de tracking. La LED 3 clignotera à chaque actualisation.

Configurer le Xbee :

Lorsque le Xbee est branché, et que l'interrupteur placé à côté est calé sur la position "Xbee", on peut brancher un câble USB sur le connecteur USB de la carte (c'est le même câble que ceux des PICKit2). Lors de la première installation, l'ordinateur détectera un composant "FT232RL". Il faudra installer son driver. Il faut ensuite lancer le logiciel X-Ctu. Sur la Figure 40, plusieurs Impressions écran ont été prises avec les paramètres à configurer pour les Xbee :

Figure 40 : Configuration des Xbee



- Vérifier la communication Modem avec "Test/Query"
- Vérifier dans la bonne communication du Xbee en tapant "+++" sans rien d'autre (ne pas appuyer sur "Entrée" non plus). Un message "OK" devrait apparaitre à côté de +++.
- Si cette communication ne fonctionne pas, modifier les paramètres du Xbee : Channel C, ID (3332), DH (0), DL (0), ainsi que le Baudrate (9600).

Le Xbee est maintenant opérationnel.

IV - Evolutions futurs :

La carte station sol V2 n'a pas pu être validée. Il faudrait donc pouvoir vérifier son bon fonctionnement. Avant de continuer à modifier la carte et le code, il serait très intéressant de pouvoir valider celle-ci entièrement. En effet, bien que certains tests aient été concluants, il faudrait poursuivre cette série de test :

Tester toutes les communications UART :

- Xbee → PIC moteur
- Xbee ↔ PIC aiguilleur
- Xbee ↔ PC USB
- PC USB → PIC Aiguilleur
- Aiguilleur → Incrustation Vidéo

Cependant, depuis peu, la communication Xbee (sur le port USB) n'est plus fonctionnelle. En priorité, il faudra donc réparer cette erreur, pour pouvoir ensuite faire les tests de communication.

Concernant le code en lui-même, celui-ci n'a jamais été testé entièrement sur la V2. Sur la carte V1, celui-ci a fonctionné assez bien. Par la suite, sur la V2, un code a été fait pour faire tourner les moteurs et ce bout de code a été validé.

Pour ce qui est de la carte puissance, celle-ci a été validée. En imposant des signaux Clock (GBF), Enable, Sens, … sur le connecteur DIL10, les moteurs ont tourné comme il fallait. Cette carte est donc validée. Des radiateurs non dimensionnés ont été posés sur les hacheurs et le régulateur 5V. Il faudra certainement en dimensionner pour les futures versions.

Lorsque la carte commande sera validée (en particulier les communications), un tirage professionnel de la carte pourra être envisagé.